Apple Assembly Line

$1.50

Volume 2 -- Issue 5 February, 1982

In This Issue...

Renew Now. the Price is Going Up

If you renew your subscription before March 1, 1982, you can renew at the current rate of $12/year. Starting March 1st, the price will go up to $15/year (2nd class mail in the USA). Subscriptions sent First Class Mail to USA, Canada, and Mexico will be $18/year. Air Mail subscriptions to all other countries will be $28/year. The price for back issues will be $1.50 each (plus $1.00 postage outside of USA, Canada, and Mexico).

S-C MACRO Assembler II is almost here!

By the time you read this, I expect to be filling orders for the new MACRO version. This is what I have been calling Version 5.0, but I have decided to call it S-C MACRO Assembler II instead. Version 4.0 will still be sold at $55. The MACRO version will be $80. Owners of Version 4.0 can upgrade for only $27.50. There will be an all new manual. rather than the current 2-part manual.

The MACRO Assembler includes macros (of course!), conditional assembly, EDIT, COPY, global string replacement, and many more new features. And it assembles even faster than version 4.0!

DOS Error Trapping from Machine Language........Lee Meador

I have been working on a text editor program for about three years now at the World Bible Translation Center. It allows us to edit in any two of the following languages: English, Russian, Greek, Hebrew, and Arabic. Hebrew and Arabic move from right to left across the screen, as they should.

Recently we have been making some enhancements to this multi-lingual text editor (called ALPHONSE) which include support of two disk drives (a program disk in drive 1 and a data disk in drive 2). But we didn't want to require the use of two drives. That means a routine must look on the various disks to see if the data is there. We can do this very handily by RENAMEing a certain file -- call it FILE -- and assuming that a DOS error means that the data isn't on that disk. Then we can look on other drives and finally, if it isn't found anywhere, we can prompt the user to put in the data disk. Then we look again -- and so on.

A problem with this is that I need to trap from assembly language any DOS errorswhich occur, but I want to return to the program if the user accidentally types the RESET key (with ALPHONSE it will always be accidentally). A second use for DOS error trapping came up because I/O errors in a disk file print the error message but do not change from the HIRES page to the text page. That makes it rather difficult to see what the error is -- especially for the less advanced user, who has no idea what is happening.

Here is a program listing of ALPHONSE with all the insides removed. Where the real program would have large sections of code, I have instead comments that look like this:

```
*->->->->->->->->->->->->->->->->->->->->
* DO SOME ACTION WHICH IS NOT SHOWN
*->->->->->->->->->->->->->->->->->->->->
```

(Of course, ALPHONSE is about 8K long and the listing is nearly 1/2-inch thick, so this isn't the whole listing.)

MAIN PROGRAM OUTLINE

Here is an outline of the main program:

```
     MACH:   GLOBAL INITIALIZATION;
     REENT:  LOCAL INITIALIZATION;
             REPEAT
               READ EDITOR COMMAND;
               PROCESS EDITOR COMMAND;
             UNTIL EDITOR COMMAND = QUIT;
     END.
```

S-C MACRO ASSEMBLER II.....$80.00
S-C ASSEMBLER II Version 4.0.....$55.00
Upgrade from Version 4.0 to MACRO.....$27.50
Includes Manual, Diskette with Assembler and sample programs, and Quick Reference Card. Call for more information.

Source code of Version 4.0 on disk.....$95.00
Fully commented, easy to understand and modify to your own tastes.

Cross Assembler Patches for 6809.....$22.50
Requires possession of Version 4.0. Enables you to develop programs for the Motorola 6809 CPU. (The MILL from Stellation, EXCEL-9 from ESD Laboratories, or the Radio Shack Color Computer.)

Cross Assembler for 6800.....$22.50
Requires possession of Version 4.0. Enables you to develop programs for the Motorola 6800, 6801, and 6802 CPUs.

AAL Quarterly Disks.....each $15.00
Each disk contains all the source code from three issues of "Apple Assembly Line", to save you lots of typing and testing time.
QD#1: Oct-Dec 1980 QD#2: Jan-Mar 1981 QD#3: Apr-Jun 1981
QD#4: Jul-Sep 1981 QD#5: Oct-Dec 1981

Double Precision Floating Point for Applesoft.....$50.00
Provides 21-digit precision for Applesoft programs.
Includes sample Applesoft subroutines for standard math functions.

FLASH! Integer BASIC Compiler (Laumer Research).....$65.00
Source Code for FLASH! Runtime Package.....$20.00

Super Disk Copy III (Sensible Software).....(reg. $30.00) $27.00

Program Line Editor (Neil Konzen, Synergistic Software).....$40.00
(Comes on DOS 3.2 disk. I have added a second disk in DOS 3.3 format with a superior set of ESCAPE macros.)

S-C Games Disk (requires Integer BASIC).....$15.00
S-C Games Disk (compiled by FLASH!, no need for Integer BASIC).....$25.00
Includes 4x4x4 tic-tac-toe, lo-res space war, lo-res jig-saw puzzle, musical memory, pentominoes, and mastermind.

Blank Diskettes.....package of 20 for $50.00
With hub rings, bulk packaged, in plain white jackets.

Lower-Case Display Encoder ROM.....$25.00
Works only Revision level 7 Apples. Replaces the encoder ROM. Comes with instructions.

Diskette Mailing Protectors.....10-99: 40 cents each
100 or more: 25 cents each
Corrugated folder specially designed for mailing mini-floppy diskettes. Fits in standard 6x9-inch envelope. (Envelopes 5-cents each, if you need them.)

Zip-Lock Bags (2-mil, 6"x9").....100 for $8.50
(2-mil, 9"x12").....100 for $13.00

Books, Books, Books.....compare our discount prices!
"Beneath Apple DOS", Worth & Lechner.....($19.95) $18.00
"What's Where in the Apple", William Leubert.....($14.95) $14.00
"6502 Assembly Language Programming", Leventhal.....($16.99) $16.00
"Apple Assembly Language", Don & Kurt Inman.....($12.95) $12.00
"MICRO on the Apple--1", includes diskette.....($24.95) $23.00
"MICRO on the Apple--2", includes diskette.....($24.95) $23.00

*** S-C SOFTWARE, P. O. BOX 280300, Dallas, TX 75228 ***
*** (214) 324-2050 We take Master Charge and VISA ***

In the global initialization we have to do four things related to error-trapping:

1. Call SETUP.DOS.TABLE to copy my addresses into the table at $9D56 of DOS. This makes DOS come back to my program when any soft entry of a funny DOS command occurs. Just calling SETUP.DOS.TABLE will not really trap any errors, but it will keep DOS from terminating your program if a DOS error does occur (that usually means SYNTAX ERROR, I/O ERROR, or FILE NOT FOUND).

2. Call CLEAR.ERROR to initialize the ONERR trapping mechanism in my program.

3. Call ON.ERROR with the address of the error-handling routine in the A and Y registers (LO, HI). This sets up the DOS error-handling capabilities as if Applesoft were running and ONERR were set.

4. After doing all the global initialization of files and such, we need to call OFF.ERROR to turn off the error handling that ON.ERROR set up. After calling OFF.ERROR any DOS error will beep and go to the soft entry point. (We have already set the soft entry point in step one to be MY.RESET.)

In the local initialization we take care of a few more things that have to be done every time the program is run -- not just the first time. The call to OFF.ERROR cleared any error trapping so we can call SETUP.DOS.TABLE and CLEAR.ERROR again without causing any problems.

Note that the call to LOOK.FOR.FILE changes the error address so we have to call ON.ERROR with MY.ERROR again to make sure that an error doesn't throw us off into never-never land. LOOK.FOR.FILE returns the carry clear if FILE is found. Carry set signals that the file isn't on any available drives; in that case, ALPHONSE would print a message like "INSERT DATA DISK AND HIT ANY KEY," then wait for a key to be pushed and call LOOK.FOR.FILE another time.

The main program loop is not really of interest here, but it is shown in the listing in skeleton form.

SUB-PROGRAMS

Now, how do the subroutines work? First, the one that you wouldn't use in your program: LOOK.FOR.FILE has to save the stack pointer. This is because we expect DOS errors to occur inside the routine. A DOS error will mess up the stack. Saving the stack lets us remember where we were. (By the way, DOS just adds things to the stack and never removes them when there is an error. The LOOK.FOR.FILE return addresses will not

Decision Systems

Decision Systems
P.O. Box 13006
Denton, TX 76203
817/382-6353

DIS-ASSEMBLER

DSA-DS dis-assembles Apple machine language programs into forms compatible with LISA, S-C ASSEMBLER (3.2 or 4.0), Apple's TOOL-KIT ASSEMBLER and others. DSA-DS dis-assembles instructions or data. Labels are generated for referenced locations within the machine language program.
$25, Disk, Applesoft (32K, ROM or Language card)

OTHER PRODUCTS

ISAM-DS is an integrated set of Applesoft routines that gives indexed file capabilities to your **BASIC** programs. Retrieve by key, partial key or sequentially. Space from deleted records is automatically reused. Capabilities and performance that match products costing twice as much.
$50 Disk, Applesoft.

PBASIC-DS is a sophisticated preprocessor for structured **BASIC**. Use advanced logic constructs such as **IF...ELSE...**, **CASE**, **SELECT**, and many more. Develop programs for Integer or Applesoft. Enjoy the power of structured logic at a fraction of the cost of **PASCAL**.
$35. Disk, Applesoft (48K, ROM or Language Card).

FORM-DS is a complete system for the definition of input and output froms. **FORM-DS** supplies the automatic checking of numeric input for acceptable range of values, automatic formatting of numeric output, and many more features.
$25 Disk, Applesoft (32K, ROM or Language Card).

UTIL-DS is a set of routines for use with Applesoft to format numeric output, selectively clear variables (Applesoft's **CLEAR** gets everything), improve error handling, and interface machine language with Applesoft programs. Includes a special load routine for placing machine language routines underneath Applesoft programs.
$25 Disk, Applesoft.

SPEED-DS is a routine to modify the statement linkage in an Applesoft program to speed its execution. Improvements of 5-20% are common. As a bonus, **SPEED-DS** includes machine language routines to speed string handling and reduce the need for garbage clean-up. Author: Lee Meador.
$15 Disk, Applesoft (32K, ROM or Language Card).

(Add $4.00 for Foreign Mail)

***Apple II is a registered trademark of the Apple Computer Co.**

be messed up.)

LOOK.FOR.FILE sets its own DOS error trap address. Then the program looks through trying to find FILE on the various slots and drives. It does this by printing the DOS commands <CTRL-D>, RENAME FILE, FILE, Sx, Dy with x and y filled in. Appropriate values for x are six, five, and seven; y would be one or two. The order in which you try the slot/drive combinations will determine which of two disks are chosen if you put two data disks in at the same time. I used a table of six slot/drive combinations to choose the order and positions to try. Notice that before printing the DOS RENAME command, I had to check to see if there was a disk card in the slot. Choosing a slot without a disk card in it for a DOS command will cause DOS to hang when you try the next DOS command with a different slot. DOS is waiting for the last drive to quit running. Little does DOS know that an empty slot always seems to be running (to DOS at least).

If the DOS RENAME command fails or there is no disk card in the slot, LOOK.FOR.FILE will jump to LOOK.ERR to loop and try the next slot/drive. If it runs out of slot/drives the program returns with carry set to indicate FILE was not found. Carry clear indicates that the last-used drive has FILE on it.

There are several routines you might want to copy as is to your program. Calling them takes care of error trapping and reset trapping.

SETUP.DOS.TABLE: copies MY.TABLE into DOS to jump to my program on any DOS error or RESET. Unfortunately, at this point you can't tell them apart.

ON.ERROR: sets the error address to the value in the A, Y (LO, HI) registers. When a DOS error occurs after ON.ERROR has been called, DOS will jump to this address with the error number in the X register. All other registers will have been changed.

OFF.ERROR: turns off the error trapping and resets DOS to the state it was in before ON.ERROR was first called. SAVE.AAB6 is used to keep track of which BASIC language DOS thinks was active. Restoring AAB6 before exiting your program will help DOS keep things sorted out. Calling OFF.ERROR restores AAB6. (By the way, while ON.ERROR is active, DOS thinks that Applesoft is currently running a program and that there has been an ONERR statement. Zero page locations $D8, $76, and $33 are used for this.)

CLEAR.ERROR: call this the first thing in your program to set up the flags used by ON.ERROR and OFF.ERROR.

Note: MY.RESET just reenters the program loop if someone types the RESET key. That makes it a null key. MY.ERROR should be looked at to see how the DOS error message comes back to you. You can use the message to print various messages depending upon what is wrong. Or, you can take various actions depending upon the error message. Pages 114-115 of the DOS manual show

what the various error numbers are that come back in the X register.

```
                         1000        .OR $803
                         1010 *-------------------------------
                         1020 *
                         1030 * ALPHONSE - MULTI-LINGUAL TEXT EDITOR
                         1040 *
                         1050 * Chopped up to show ERROR trapping
                         1060 *  ala Applesoft ONERR command.
                         1070 *  NOTE: There is no RESUME but
                         1080 *  you are able to easily pick
                         1090 *  up DOS errors and handle them
                         1100 *  while disabling the RESET
                         1110 *  (on AutoStart ROM).
                         1120 *
                         1130 * by Lee Meador
                         1140 *
                         1150 *      MACH - Main program entry
                         1160 *      REENT- Program re-entry
                         1170 *      ULOOP- Main program loop
                         1180 *      MY.RESET- handle RESET key pushed
                         1190 *      MY.ERROR- default error handler
                         1200 *      END  - Exit to BASIC
                         1210 *      SETUP.DOS.TABLE- hook in RESET trapping
                         1220 *      ON.ERROR - set error trap
                         1230 *      OFF.ERROR- kill error trap
                         1240 *      CLEAR.ERROR- init error flags
                         1250 *      LOOK.FOR.FILE- find S,D of FILE
                         1260 *      MY.TABLE - copied into DOS table
                         1270 *
                         1280 *-------------------------------
9D56-                    1290 DOS.TABLE .EQ $9D56
FC58-                    1300 HOME.TEXT .EQ $FC58
0000-                    1310 TMP1      .EQ 0       PAGE 0
                         1320 *-------------------------------
                         1330 *
                         1340 * THIS IS THE MAIN ENTRY POINT
                         1350 *  FOR ALPHONSE.
                         1360 *
                         1370 *-------------------------------
0803- 20 44 08           1380 MACH       JSR SETUP.DOS.TABLE
0806- 20 79 08           1390            JSR CLEAR.ERROR  NO ONERR
0809- A9 3B              1400            LDA #MY.ERROR    THEN SET IT
080B- A0 08              1410            LDY /MY.ERROR    .. TO MY.ERROR
080D- 20 50 08           1420            JSR ON.ERROR
0810- 20 58 FC           1430            JSR HOME.TEXT CLR TXT SCR
                         1440 *->->->->->->->->->->->->->->->->
                         1450 * DO INITIALIZE PROCESSING
                         1460 *->->->->->->->->->->->->->->->->
0813- 20 70 08           1470            JSR OFF.ERROR ON ERR TURNED OFF
                         1480 *-------------------------------
                         1490 * RE-ENTRY POINT. NORMAL ENTRY COMES HERE TOO
                         1500 *-------------------------------
0816- 20 44 08           1510 REENT      JSR SETUP.DOS.TABLE
0819- 20 79 08           1520            JSR CLEAR.ERROR NO ON ERR
081C- 20 82 08           1530 .10        JSR LOOK.FOR.FILE
081F- 90 0D              1540            BCC LOAD.FILE
0821- A9 3B              1550            LDA #MY.ERROR     SET ERROR
0823- A0 08              1560            LDY /MY.ERROR     .. TO MY.ERROR
0825- 20 50 08           1570            JSR ON.ERROR
0828- 20 58 FC           1580            JSR HOME.TEXT CLEAR SCRN
                         1590 *->->->->->->->->->->->->->->->->
                         1600 * PRINT "INSERT CORRECT DISK"
                         1610 *->->->->->->->->->->->->->->->->
082B- 4C 1C 08           1620            JMP .10       TRY AGAIN TO FIND TEXT.DIR
                         1630 *-------------------------------
                         1640 LOAD.FILE
082E- A9 3B              1650            LDA #MY.ERROR FIX ERROR HANDLER
0830- A0 08              1660            LDY /MY.ERROR
0832- 20 50 08           1670            JSR ON.ERROR
                         1680 *->->->->->->->->->->->->->->->->
                         1690 * THE REST OF INITIALIZING
                         1700 *->->->->->->->->->->->->->->->->
                         1710
```

```
                  ULOOP
                  *->->->->->->->->->->->->->->->->->
                  * MAIN PROGRAM LOOP DOES EACH
                  *  COMMAND TYPED
                  * EXIT COMMAND JUMPS TO "END"
                  *->->->->->->->->->->->->->->->->->
0835- 4C 35 08           JMP ULOOP    .. LOOP IF UNDEF
                  *--------------------------------
                  *
                  * ROUTINE TO HANDLE USER HITTING
                  *  RESET. (HANDLED IN DOS--DOS
                  *  FIXES IT ON $3D3 EXIT.)
                  * NO HOOKS TO CHANGE AND FIX BACK
                  *
                  *--------------------------------

                  MY.RESET
                  *->->->->->->->->->->->->->->->->->
                  * RESET POINTERS AND HIRES PAGE2
                  *->->->->->->->->->->->->->->->->->
0838- 4C 35 08           JMP ULOOP

                  *--------------------------------
                  *
                  * MY GENERAL ERROR HANDLER JUST
                  *  PRINTS "ERROR NUMBER " AND
                  *  THE NUMBER FOR THE ERROR THEN
                  *  EXITS TO WHATEVER BASIC WAS
                  *  RUNNING BEFORE.
                  *
                  *--------------------------------
                  MY.ERROR
083B- 8A                 TXA          SAVE ERR NUM
083C- 48                 PHA
                  *->->->->->->->->->->->->->->->->->
                  * HOME SCREEN AND PRINT THE
                  *  MESSAGE "ERROR NUMBER "
                  *->->->->->->->->->->->->->->->->->
083D- 68                 PLA          ERR NUMBER
                  *->->->->->->->->->->->->->->->->->
                  * PRINT ACC AS DECIMAL NUMBER
                  *  FOLLOWED BY A <RETURN>
                  *->->->->->->->->->->->->->->->->->
083E- 20 70 08    END    JSR OFF.ERROR FIX UP $AAB6:
0841- 4C D3 03           JMP $3D3      HARD EXIT RESTORS DOS.TABLE
                  *--------------------------------
                  *
                  * COPY MY ADDRESSES INTO THE DOS
                  *  TABLE OF JUMPS (AT $9D56).
                  *
                  *--------------------------------
                  SETUP.DOS.TABLE
0844- A2 0C              LDX #12      12 BYTES
0846- BD C6 08    .10    LDA MY.TABLE-1,X
0849- 9D 55 9D           STA DOS.TABLE-1,X
084C- CA                 DEX
084D- D0 F7              BNE .10
084F- 60                 RTS
                  *--------------------------------
                  *
                  * DOS ERROR SETUP/RESET
                  *
                  *  CALL CLEAR.ERROR AT START OF
                  *   PROGRAM TO SET UP FLAG
                  *   (ITS ALSO OK AFTER OFF.ERROR)
                  *  CALL ON.ERROR WITH A,Y HOLDING
                  *   THE ADDRESS YOU WANT TO JUMP
                  *   TO IF A DOS ERROR OCCURS.
                  *  CALL OFF.ERROR TO CANCEL ERROR
                  *   TRAPPING AND REVERT TO NORMAL
                  *   ERROR MSG AND JUMP TO BASIC
                  *
                  *  WHEN THE ERROR ROUTINE IS
                  *   CALLED (ON AN ERROR) THE X
                  *   REGISTER HOLDS THE ERROR
                  *   NUMBER AS LISTED P 114-115 OF
                  *   THE DOS MANUAL.
                  *
```

```
                 *  AN ERROR WILL CAUSE THE STACK
                 *   TO BE MESSED UP. SO, SAVE IT
                 *   WHEN YOU EXPECT ERRORS.
                 *
                 *--------------------------------
                 ON.ERROR
0850- 8D 5A 9D          STA DOS.TABLE+4
0853- 8C 5B 9D          STY DOS.TABLE+5
0856- AD 81 08          LDA SAVE.AAB6     48K ONLY
0859- D0 07             BNE .10
085B- AE B6 AA          LDX $AAB6         48K ONLY !!!!!
085E- CA                DEX
085F- 8E 81 08          STX SAVE.AAB6     48K ONLY
0862- A9 40      .10    LDA #$40     PRETEND AS(])
0864- 8D B6 AA          STA $AAB6         48K ONLY
0867- 0A                ASL          $80
0868- 85 D8             STA $D8      ONERR ACTIVE
086A- 0A                ASL          $00
086B- 85 76             STA $76      AS(]) RUNNING
086D- 85 33             STA $33       (REALLY)
086F- 60                RTS
                 OFF.ERROR
0870- AE 81 08          LDX SAVE.AAB6
0873- F0 04             BEQ CLEAR.ERROR   ZERO->NEVER SET
0875- E8                INX
0876- 8E B6 AA          STX $AAB6         48K ONLY
                 CLEAR.ERROR
0879- A9 00             LDA #0            CLEAR FLAGS
087B- 8D 81 08          STA SAVE.AAB6
087E- 85 D8             STA $D8           CLEAR ONERR FLAG
0880- 60                RTS
                 *--------------------------------
0881- 00         SAVE.AAB6 .HS 00    FLAG
                 *--------------------------------
                 *
                 * LOOK FOR FILE ON VARIOUS DRIVES
                 *
                 *  RETURNS CARRY CLEAR IF FOUND
                 *   AND SET IF NOT. USES RENAME
                 *   FILE,FILE TO SEE IF FILE EXISTS
                 *
                 *--------------------------------
                 LOOK.FOR.FILE
0882- BA                TSX          SAVE STACK
0883- 8E BF 08          STX LOOK.STACK
0886- A9 B7             LDA #LOOK.ERR
0888- A0 08             LDY /LOOK.ERR
088A- 20 50 08          JSR ON.ERROR
088D- A9 00             LDA #0       TABLE OFFSET
088F- 8D BE 08          STA LOOK.CNT
                 *--------------------------------
                 LOOK.LOOP
0892- AE BE 08          LDX LOOK.CNT
0895- EC BD 08          CPX LOOK.MAX (# OF TRYS)
0898- B0 18             BCS .99      FAIL EXIT
                 *->->->->->->->->->->->->->->->->->
                 * CHECK FOR DISK CARD IN SLOT
                 *  SO THINGS WON'T HANG. FIRST,
                 *  LOAD THE ACC WITH THE SLOT
                 *  THEN ...
                 *->->->->->->->->->->->->->->->->->

089A- 29 07             AND #$07     SLOT #
089C- 09 C0             ORA #$C0
089E- 85 01             STA TMP1+1
08A0- A9 00             LDA #0
08A2- 85 00             STA TMP1     TMP1=CS00
                 * TMP1 IS SLOT ADDRESS
                 * CHECK BYTES 7,5,3,1 FOR MATCH
                 *  AS AUTO MONITOR DOES
08A4- A0 07             LDY #$07     SAME AS MONITOR ($FABA AUTO)
08A6- B1 00      .10    LDA (TMP1),Y FETCH SLOT BYTE
08A8- D9 BF 08          CMP DISKID-1,Y  IS IT DISK?
08AB- D0 0A             BNE LOOK.ERR NOPE...
08AD- 88                DEY          DOWN TWO
08AE- 88                DEY
08AF- 10 F5             BPL .10      AND LOOP
```

```
                    3270 * THERE IS A DISK CARD THERE
                    3280 *-------------------------------
                    3290 *->->->->->->->->->->->->->->->->
                    3300 * PRINT OUT CTRL-D THEN "RENAME
                    3310 *  FILE,FILE,SX,DX" FILLING IN
                    3320 *  X ACCORDING TO THE VALUE OF
                    3330 *  LOOK.CNT
                    3340 *->->->->->->->->->->->->->->->->
08B1- 18            3350         CLC           FOUND IT
08B2- AE BF 08      3360 .99     LDX LOOK.STACK
08B5- 9A            3370         TXS           RESTORE STACK
08B6- 60            3380         RTS
                    3390 *-------------------------------
                    3400 * COME HERE IF DOS COMMAND FAILS
                    3410 *-------------------------------
                    3420 LOOK.ERR
08B7- EE BE 08      3430         INC LOOK.CNT
08BA- 4C 92 08      3440         JMP LOOK.LOOP
                    3450 *-------------------------------
08BD- 06            3460 LOOK.MAX    .DA #6
08BE-               3470 LOOK.CNT    .BS 1
08BF-               3480 LOOK.STACK  .BS 1
                    3490 *-------------------------------
08C0- 20 FF 00
08C3- FF 03 FF
08C6- 3C            3500 DISKID      .HS 20FF00FF03FF3C MATCHES DISK CARD+1 TO 7
                    3510 *-------------------------------
                    3520 *
                    3530 * TABLE OF ERR/RESET ADDRESSES
                    3540 *
                    3550 *-------------------------------
08C7- 38 08         3560 MY.TABLE .DA MY.RESET
08C9- 38 08         3570          .DA MY.RESET
08CB- 3B 08         3580          .DA MY.ERROR
08CD- 00 E0         3590          .DA $E000
08CF- 38 08         3600          .DA MY.RESET
08D1- 38 08         3610          .DA MY.RESET
                    3620         .EN
```

Improving the Epson Controller.....Peter G. Bartlett, Jr.

[I recently bought an NEC PC-8023 dot matrix printer, which has fabulous features. The store sold me an Epson controller to run it, assuring me it was all I needed. Naturally, they were wrong. To get all features, just as with the Epson printer, you need to be able to send 8-bit characters. I figured out how, and was just about to write an article about it, when the following one came from Peter Bartlett of Chicago, Illinois. (Bob Sander-Cederlof)]

As you may know. the Epson MX-80 printer is somewhat hamstrung by the Epson Controller. Certain features, such as the "TRS-80" graphics character set, are not available. You can buy the Graftrax kit to enable dot graphics, but these built-in character graphics are still inaccessible.

The problem is in the card Epson makes to interface its line of printers with the Apple. (The problem is not present if you use a non-Epson card.) Hardware on the Epson controller card masks out the high-order bit, eliminating the ability to access the standard graphics characters and some of the dot graphics capabilities.

Epson's reasoning is that the Apple only sends characters with the high-bit set, so Epson has hardware on the card to mask out that bit. That way the normal characters print as they should.

If Epson had masked out the high bit in their printer driver routine instead, then machine language programmers like us could have accessed all the features of the printer. We could bypass the printer driver and work directly with the printer I/O port.

Fortunately, the card has jumpers that can be changed and the printer driver is on an EPROM that can be changed.

So you will need a soldering iron, an EPROM blaster, and an erased 2708 EPROM. [The EPROM Blaster from Apparat can blow 2708's. I don't believe the Mountain Hardware ROMWRITER can.]

On the Epson interface card, the jumper marked "P4" should be removed and installed on "M4" instead. This jumper are directly underneath the EPROM and are labelled. [Ignore the three jumpers on the right side of the EPROM.] This fix is not documented, although you can see it in the Schematic Drawing of the card. I called Epson on the phone, and they told me about it. With the jumper moved to M4, the high-bit is transmitted correctly.

BUT!!! Now normal characters do not print normally! Instead, you get the graphics characters! Okay, we need to modify the program inside the EPROM. At location $C120 (assuming the card is in slot 1) you will find an instruction "AND #$7F". This clears the high-bit for processing control codes only. We need to move this instruction to $C112, so that the hihg-bit is cleared for transmitted codes also. Here is a listing of the BEFORE and AFTER programs, with the moved instruction starred.

```
                        1000 *--------------------------------
                        1010 *      CHANGES TO EPSON CONTROLLER 2708 ROM
                        1020 *--------------------------------
                        1030 *---AS IT NOW IS-----------------
                        1040         .OR $C111
                        1050         .TA $811
C111- 68                1060         PLA
C112- A8                1070         TAY
C113- CA                1080         DEX
C114- 9A                1090         TXS
C115- 68                1100         PLA
C116- 28                1110         PLP
C117- AA                1120         TAX
C118- 90 38             1130         BCC $C152
C11A- BD B8 05          1140         LDA $5B8,X
C11D- 10 19             1150         BPL $C138
C11F- 98                1160         TYA
C120- 29 7F *           1170      *  AND #$7F  *  STRIP OFF SIGN
C122- 49 30             1180         EOR #$30
                        1190 *---AS IT NEEDS TO BE------------
                        1200         .OR $C111
                        1210         .TA $811
C111- 68                1220         PLA
C112- 29 7F *           1230      *  AND #$7F  *  STRIP OFF SIGN BIT
C114- A8                1240         TAY
C115- CA                1250         DEX
C116- 9A                1260         TXS
C117- 68                1270         PLA
C118- 28                1280         PLP
C119- AA                1290         TAX
C11A- 90 36             1300         BCC $C152
C11C- BD B8 05          1310         LDA $5B8,X
C11F- 10 17             1320         BPL $C138
C121- 98                1330         TYA
C122- 49 30             1340         EOR #$30
```

That fix in the program will clear the high-bit off every character sent via the printer driver to the printer. We are back where we started. Except that now the clever programmer can send characters directly to the printer, bypassing the EPROM resident driver. Here is how to send one character directly to the printer:

```
OUTPUT STA $C090   Assuming slot 1
.1     BIT #C1C1   Character picked up by printer?
       BMI .1      No, keep testing
```

I have tried everything above, and it all works perfectly. I hope it proves useful to lots of you AAL readers.

Even Faster Primes.....Charles Putney
[Charlie is a long-time friend and subscriber in Ireland]

Bob, I wanted to answer your challenge in the Ocotber 1981 AAL for some time, but this is the first chance I had. You sifted out the primes in 690 milliseconds, and challenged readers to beat your time. I did it!

I increased the speed by using a faster algorithm, and by using some self-modifying code in the loops. I know self-modifying code is dangerous, and a NO-NO, but it amounts to about 50 milliseconds improvement.

The algorithm changes are an even greater factor. The main ideas for the sieve are:

1. Only check odd numbers
2. Get next increment from the prime array.
   This means you only knock out primes.
3. Start knocking out at P^2. That is,
   if prime found is 3. start at 9.
4. Increment the knock-out index by 2*P.
   This avoids knocking out even numbers.
5. Stop at the square-root of the maximum number.

Your algorithm did all the above except 3 and 4.

With these routines, a generation takes 330 milliseconds. This is over twice as fast as yours!

You could still shave a little time off by optimizing the square routine, and even including it inline since it is only called from one place.

I'll grant you that this is not the same algorithm, but the goal is to find primes fast. I know throw down the glove for the next challenger!

```
10  TEXT : HOME : PRINT "CHARLES PUTNEY'S FASTER PRIME GENERATOR
    ----------------------------------"
20  VTAB 10: HTAB 15: PRINT "LOADING . . ."
30  HGR : TEXT
40 D$ =  CHR$ (4): PRINT D$"BLOAD B.PUTNEY'S PRIMES"
50  HOME : VTAB 10: HTAB 10: PRINT "HIT ANY KEY TO START"
60  POKE 49168,0: GET A$: POKE 49168,0
80  POKE 49232,0: POKE 49239,0
90  CALL 32768
95  TEXT : FOR A = 8195 TO 24576 STEP 2: IF  PEEK (A) = 0 THEN  PRINT
    A - 8192;" ";
98  NEXT
100  REM   PRIME TESTER
110  REM   CHARLES H. PUTNEY
120  REM   18 QUINNS ROAD
130  REM   SHANKILL
140  REM   CO. DUBLIN
150  REM   IRELAND
160  REM   TIME FOR 100 RUNS = 42 SECONDS
```

```
                      1000          .OR $8000     SAFELY OUT OF WAY
                      1010          .TF B.PUTNEY'S PRIMES
                      1020 *--------------------------------
2000-                 1030 BASE     .EQ $2000     BASE OF PRIME ARRAY
FF3A-                 1040 BEEP     .EQ $FF3A     BEEP THE SPEAKER
                      1050 *--------------------------------
                      1060 *        MAIN CALLING ROUTINE
                      1070 *
8000- A9 64           1080 MAIN     LDA #100      DO 100 TIMES SO WE CAN MEASURE
8002- 8D BA 80        1090          STA COUNT     THE TIME IT TAKES
8005- 20 3A FF        1100          JSR BEEP      ANNOUNCE START
8008- 20 1C 80        1110 .1       JSR ZERO      CLEAR ARRAY
800B- A9 03           1120          LDA #$03
800D- 8D B8 80        1130          STA START     SET STARTING VALUE
8010- 20 39 80        1140          JSR PRIME
8013- CE BA 80        1150          DEC COUNT     CHECK COUNT
8016- D0 F0           1160          BNE .1        DONE ?
8018- 20 3A FF        1170          JSR BEEP      SAY WE'RE DONE
801B- 60              1180          RTS
                      1190 *--------------------------------
                      1200 *        ROUTINE TO ZERO MEMORY
                      1210 *        FROM $2000 TO $6000
                      1220 *
801C- A9 01           1230 ZERO     LDA #BASE+1   START AT $2001
801E- 8D 2C 80        1240          STA .1+1      MODIFY OUR STORE
8021- A9 20           1250          LDA /BASE+1
8023- 8D 2D 80        1260          STA .1+2
8026- A9 00           1270          LDA #$00      GET A ZERO
8028- AA              1280          TAX           SET INDEX
8029- A0 40           1290          LDY #$40      NUMBER OF PAGES
802B- 9D FF FF        1300 .1       STA $FFFF,X   MODIFIED AS WE GO
802E- E8              1310          INX           EVERY ODD LOCATION
802F- E8              1320          INX
8030- D0 F9           1330          BNE .1        NOT DONE
8032- EE 2D 80        1340          INC .1+2      NEXT PAGE
8035- 88              1350          DEY
8036- D0 F3           1360          BNE .1        NOT YET
8038- 60              1370          RTS
                      1380 *--------------------------------
                      1390 *        PRIME ROUTINE
                      1400 *        SETS ARRAY STARTING AT BASE
                      1410 *        TO $FF IF NUMBER IS NOT PRIME
                      1420 *        CHECKS ONLY ODD NUMBERS > 3
                      1430 *        INC = INCREMENT OF KNOCKOUT
                      1440 *        N = KNOCKOUT VARIABLE
                      1450 *
8039- AD B8 80        1460 PRIME    LDA START
803C- 0A              1470          ASL           INC = START * 2
803D- 8D B9 80        1480          STA INC
8040- 20 7F 80        1490          JSR SQUARE    SET N = N * N
8043- 18              1500          CLC           ADD BASE TO N
8044- AD 5B 80        1510          LDA N+1
8047- 69 00           1520          ADC #BASE
8049- AA              1530          TAX           KEEP LOW ORDER PART IN X
804A- A9 00           1540          LDA #0        N+1 TO ZERO
804C- 8D 5B 80        1550          STA N+1
804F- AD 5C 80        1560          LDA N+2
8052- 69 20           1570          ADC /BASE
8054- 8D 5C 80        1580          STA N+2
8057- A8              1590          TAY
8058- A9 FF           1600 LOOP     LDA #$FF      FLAG AS NOT PRIME
805A- 9D FF FF        1610 N        STA $FFFF,X   REMEMBER THAT N IS REALLY AT N+1
805D- 18              1620          CLC           N = N + INC
805E- 8A              1630          TXA           N=N+INC
805F- 6D B9 80        1640          ADC INC
8062- AA              1650          TAX
8063- 90 F3           1660          BCC LOOP      DONT'T BOTHER TO ADD, NO CARRY
8065- C8              1670          INY           INC HIGH ORDER
8066- 8C 5C 80        1680          STY N+2
8069- C0 60           1690          CPY /BASE+$4000  IF IS GREATER THAN $6000
806B- 90 EB           1700          BCC LOOP      NO, REPEAT
806D- AE B8 80        1710          LDX START     GET OUR NEXT KNOCKOUT
8070- E8              1720 .1       INX
8071- E8              1730          INX           START = START + 2
8072- 30 0A           1740          BMI .2        WE'RE DONE IF X>$7F
8074- BD 00 20        1750          LDA BASE.X    GET A POSSIBLE PRIME
8077- D0 F7           1760          BNE .1        THIS ONE HAS BEEN KNOCKED OUT
8079- 8E B8 80        1770          STX START
807C- F0 BB           1780          BEQ PRIME     ...ALWAYS
807E- 60              1790 .2       RTS
```

```
                        1800 *--------------------------------
                        1810 *       SQUARE ROUTINE
                        1820 *       TAKES SQUARE OF NUMBER
                        1830 *       IN START (ONE BYTE) AND
                        1840 *       PUTS RESULT IN N+1 (LOW)
                        1850 *       AND N+2 (HIGH)
                        1860 *
807F- A9 00     1870 SQUARE LDA #$00
8081- 8D 5B 80  1880        STA N+1       CLEAR N
8084- 8D 5C 80  1890        STA N+2
8087- 8D BC 80  1900        STA MULT+1    AND MULTIPLIER HIGH
808A- AD B8 80  1910        LDA START
808D- 8D BB 80  1920        STA MULT      MULT LOW = START
8090- 8D BD 80  1930        STA SHCNT     SHIFT COUNTER
8093- A2 08     1940        LDX #$08      EIGHT SHIFTS
8095- 6E BD 80  1950 .1     ROR SHCNT     GET LS BIT IN CARRY
8098- 90 13     1960        BCC .2        DON'T ADD THIS TIME
809A- 18        1970        CLC           N = N + MULT
809B- AD 5B 80  1980        LDA N+1
809E- 6D BB 80  1990        ADC MULT
80A1- 8D 5B 80  2000        STA N+1
80A4- AD 5C 80  2010        LDA N+2
80A7- 6D BC 80  2020        ADC MULT+1
80AA- 8D 5C 80  2030        STA N+2
80AD- 18        2040 .2     CLC           SHIFT MULT (BOTH BYTES)
80AE- 2E BB 80  2050        ROL MULT
80B1- 2E BC 80  2060        ROL MULT+1
80B4- CA        2070        DEX
80B5- D0 DE     2080        BNE .1        MORE BITS ?
80B7- 60        2090        RTS
80B8- 00        2100 START  .DA #*-*      STARTING KNOCKOUT
80B9- 00        2110 INC    .DA #*-*      INCREMENT FOR KNOCKOUT
80BA- 00        2120 COUNT  .DA #*-*      COUNT FOR 100 TIMES LOOP
80BB- 00 00     2130 MULT   .DA *-*       MULTIPIER
80BD- 00        2140 SHCNT  .DA #*-*      SHIFT COUNT MULTIPLIER
```

Printer Handler with FIFO Buffer.....Jim Kassel
[Jim Kassel is a subscriber from St. Paul, Minnesota.]

Before I get on with technical discussions, first let me say that I have had a ball using S-C Assembler II Version 4.0. It definitely has earned a place on the list of "The Greatest Things Since Sliced Bread." My current version incorporates the block move and copy feature described in the December '80 and January '81 issues of AAL which have been a welcome enhancement.

Now...on with the article, about a super simple programming technique that I have used extensively. I am a hardware logic designer by trade and before the introduction of First In-First Out (FIFO) memory chips, designers had to implement that function using an input address up-counter, an output address up-counter, and an up/down counter to determine character count. Now that the FIFO chips are available, they are still a bit expensive for home computer use. By using the old counter method implemented in software, not only is the FIFO free but also extrememly expandable in size (within the bounds of the computer memory, of course).

I am going to give a little background into the necessity, in my case, for using this technique. I feel that the problem I experienced may be interesting reading to others who may have had similar occurrences.

I was writing an assembly language program that would allow my Apple II to become a terminal, using the Hayes Micromodem II and Epson MX-80 printer/Orange Micro Grappler interface card.

For the sake of versatility I would have preferred to perform operations like JSR $Cx00 (x = slot number) when transferring data with these devices. However, it became apparent that I would have to bypass the firmware on the other interface cards. This was especially true with the printer interface card. Because the printer takes 1-2 seconds to print out a line of characters, the interface becomes unavailable for storage. Since the modem wants to supply characters at a rate of up to 30 cps, at least that many characters were being "dropped on the floor" while the printer interface card kept program control.

I finally had to get the schematics and/or firmware disassemblies of the other interface cards. From them I figured out the addresses of, and the methods of communications for, the various control, data, and (most important) status registers. This allowed me to check for printer busy, modem transmit register not yet empty and modem receive register not yet full. Now I could do other things when no data could be transferred. No longer would I have to be a slave to the equipment that is used for support!

The only other problem, then, was to be able to save the print characters in a FIFO print buffer so they would not be forgotten while the printer was busy printing the previous line

of characters. In my version I allow a whole page of memory ($94) to be used for the buffer space. As long as there is not a horribly long burst of received carriage returns (the slowest printer operation), 256 locations is more than adequate because the MX-80 prints at least twice as fast as the modem data rate. Plus non-control characters are transferred into the printer line buffer much faster than incoming modem characters and the FIFO almost always stays empty because of this.

As characters arrive from the modem they are placed into the FIFO (by executing a JSR PRINT.FIFO.INPUT), then the input index (PBII) and the character counter (PBCC) are incremented. Whenever the program is in a wait loop (keyboard entries, modem data transfers, etc.) there are no less than 33 milliseconds (300 baud/30 cps) to do non-critical operations. This is more than enough time to execute a JSR PRINT.FIFO.OUTPUT.1 instruction during each "round trip" of the wait loops. If the printer is busy. the program is returned to with no data transferred; if the printer is not busy, the program is returned to after the next FIFO output character is sent, the output index (PBOI) is incremented, and the character counter (PBCC) is decremented. In any case. the program does not depend on the outcome of the subroutine results. The subroutines maintain their independence by correctly updating and monitoring the character counter (PBCC).

In my version, I must append a line feed (<LF>) character to every carriage return (<CR>) that is sent. I check every FIFO input character to see if it is a <CR>. If so, I store a <LF> into the next FIFO input location. Note that if I had decided to send the <LF> directly to the printer by monitoring for the <CR> in the FIFO output subroutine. I would again have been a slave to the printer while waiting for it to become unbusy with the <CR> operation.

By making PRINT.FIFO.OUTPUT.2 a separate subroutine. I could write it for any printer interface card with data and status registers and still not require any changes to subroutines PRINT.FIFO.INPUT and PRINT.FIFO.OUTPUT.1. This provided some versatility for converting the program for some friends with different interfaces.

```
                1000 *--------------------------------
                1010 *          PRINTER HANDLER
                1020 * USED SO THAT PROGRAM DOESN'T HANG
                1030 *      WHEN PRINTER IS BUSY
                1040 *
                1050 *           JIM KASSEL
                1060 *       1161 GOODRICH AVE.
                1070 *       ST. PAUL, MN 55105
                1080 *--------------------------------
   0010-        1090 PRINT.SLOT.SHIFTED  .EQ $10
                1100 *                   PRINTER SLOT # SHIFTED LEFT BY 4
   00CE-        1110 PBII    .EQ $CE     PRINT BUFF INPUT INDEX
   00CF-        1120 PBOI    .EQ $CF     PRINT BUFF OUTPUT INDEX
   001F-        1130 PBCC    .EQ $1F     PRINT BUFF CHAR COUNT
   9400-        1140 PBUFF   .EQ $9400   PRINT BUFF BASE ADDRESS
   000D-        1150 CR      .EQ $D      CARRIAGE RETURN WITH MSB CLR
   000A-        1160 LF      .EQ $A      LINE FEED WITH MSB CLR
                1170 *--------------------------------
```

```
0800-                1180 START  .EQ $800
                     1190        .OR START
                     1200 *--------------------------------
                     1210 * PRINT BUFF INPUT SUBROUTINE
                     1220 *--------------------------------
                     1230 PRINT.FIFO.INPUT
0800- 48             1240        PHA
0801- 29 7F          1250        AND #$7F      CLEAR BIT 7
                     1260
0803- A4 CE          1270 .1     LDY PBII
0805- 99 00 94       1280        STA PBUFF,Y   STORE CHAR IN PRINT BUFF
0808- E6 CE          1290        INC PBII      INCREMENT INPUT INDEX
080A- E6 1F          1300        INC PBCC      INCREMENT CHAR COUNT
                     1310
080C- C9 0D          1320        CMP #CR       CARRIAGE RETURN?
080E- D0 04          1330        BNE .2        NO
0810- A9 0A          1340        LDA #LF       YES
0812- D0 EF          1350        BNE .1        SEND <LF>
                     1360
0814- 68             1370 .2     PLA           RESTORE CHAR
0815- 60             1380        RTS
                     1390 *--------------------------------
                     1400 * PRINTER OUTPUT SUBROUTINE
                     1410 *--------------------------------
                     1420 PRINT.FIFO.OUTPUT.1
0816- A5 1F          1430        LDA PBCC      PRINT BUFF EMPTY?
0818- F0 0E          1440        BEQ .1        YES
                     1450
081A- A4 CF          1460        LDY PBOI      NO
081C- B9 00 94       1470        LDA PBUFF,Y   GET PRINT CHAR
081F- 20 29 08       1480        JSR PRINT.FIFO.OUTPUT.2
                     1490 *                       HANDLER OF SPECIFIC INTERFACE
0822- B0 04          1500        BCS .1        DON'T UPDATE IF PRINTER WAS BUSY
                     1510
0824- E6 CF          1520        INC PBOI      ELSE, INCREMENT OUTPUT INDEX
0826- C6 1F          1530        DEC PBCC      AND DECREMENT CHAR COUNT
                     1540
0828- 60             1550 .1     RTS
                     1560 *--------------------------------
                     1570 *   HANDLER FOR THE GRAPPLER (+)
                     1580 *         INTERFACE CARD
                     1590 *      AND MX-80 PRINTER(++)
                     1600
                     1610 * PRINT CHAR MUST BE IN THE A-REG
                     1620 *   CARRY SET IF CHAR NOT SENT
                     1630 *   CARRY CLEARED IF CHAR SENT
                     1640
C081-                1650 PSTAT  .EQ $C081     PRINTER STATUS REG
C081-                1660 PREG   .EQ $C081     PRINTER DATA REG
C082-                1670 PSTRBL .EQ $C082     PRINTER STROBE LOW
C084-                1680 PSTRBH .EQ $C084     PRINTER STROBE HIGH
                     1690 *--------------------------------
                     1700 PRINT.FIFO.OUTPUT.2
0829- AA             1710        TAX           SAVE PRINT CHAR
082A- A4 10          1720        LDY PRINT.SLOT.SHIFTED
082C- B9 81 C0       1730        LDA PSTAT,Y   GET PRINTER STATUS
082F- 29 0A          1740        AND #$A       MASK
0831- 49 02          1750        EOR #$2       PRINTER SELECTED AND NOT BUSY?
0833- D0 0D          1760        BNE .1        NO. EXIT
                     1770
0835- 8A             1780        TXA           YES. RESTORE PRINT CHAR
0836- 99 81 C0       1790        STA PREG,Y    LOAD PRINTER OUTPUT REG
0839- 99 82 C0       1800        STA PSTRBL,Y  SET STROBE
083C- 99 84 C0       1810        STA PSTRBH.Y  CLR STROBE
083F- 18             1820        CLC           CLEAR CARRY
0840- 90 01          1830        BCC .2        EXIT
0842- 38             1850 .1     SEC           SET CARRY
                     1860
0843- 60             1870 .2     RTS
                     1880 *--------------------------------
                     1890 END
0044-                1900 SIZE   .EQ END-START
                     1910 *--------------------------------
                     1920 * NOTE:
                     1930 * (+) : TRADEMARK OF ORANGE MICRO, INC.
                     1940 * (++): TRADEMARK OF EPSON AMERICA, INC.
                     1950 *--------------------------------
```

Patches for Applewriter to Unhook PLE.....Bob Sander-Cederlof

If you use Applewriter a lot, like I do.... And if you use Neil Konzen's Program Line Editor (PLE) a lot, like I do.... Then you probably have at least once tried to BRUN TEDITOR while PLE was still installed, like I have....

The result is maddening, to say the least. Everything seems fine. You can load a file into Applewriter, or enter a new one. You can edit to your hearts content. Then you try to SAVE it on disk. POW! What happened?!! Since PLE is still hooked into DOS, it needs to remain unmolested in memory. But Applewriter ignores its presence, and puts the text right over the top of it.

I thought I had finally learned my lesson, but then I did it again!

Finally, I decided to make Applewriter unhook everything that PLE might have hooked in, during initialization. It turned out to be surprisingly easy. Here are the patches. I haved moved them up high enough so that if you have the lower case patches installed there is no conflict. Now I do not need to reboot to get rid of PLE.

```
                    1000 *--------------------------------
                    1010 * APPLEWRITER PATCH TO UNHOOK PLE
                    1020 *--------------------------------
                    1030         .OR $803
                    1040         .TF AW.1
0803- 20 73 18      1050         JSR PATCH     REPLACES "JSR $10F8"
                    1060 *--------------------------------
                    1070         .OR $1873     SAFE PATCH AREA
                    1080         .TF AW.2
1873- 20 89 FE      1090 PATCH   JSR $FE89     SET INPUT TO KEYBOARD
1876- 20 93 FE      1100         JSR $FE93     SET OUTPUT TO SCREEN
1879- A9 9C         1110         LDA #$9C
187B- 8D 01 9D      1120         STA $9D01     RESTORE NORMAL DOS BUFFERS
187E- A9 03         1130         LDA #3
1880- 8D 57 AA      1140         STA $AA57     MAXFILES=3
1883- 20 D4 A7      1150         JSR $A7D4
1886- A2 2F         1160         LDX #$2F
1888- BD 51 9E      1170 .1      LDA $9E51,X   RESTORE PAGE 3 POINTERS
188B- 9D D0 03      1180         STA $3D0.X
188E- CA            1190         DEX
188F- 10 F7         1200         BPL .1
1891- 20 EA 03      1210         JSR $3EA      RE-HOOK DOS
1894- 4C F8 10      1220         JMP $10F8     DO WHAT THE "JSR PATCH" COVERED
```

Great Adventure.....Jeff Jacobsen

Have you ever played the ORIGINAL game? Adventure game, that is? Adventure was originally developed by Willie Crowther and Don Woods in FORTRAN on a DEC PDP-10 computer. It is the grandfather, or maybe great-grandfather by now, of the hundreds of Adventure games you see in the advertisements (you might even have bought some!).

I used the S-C Assembler II to write an Apple version of the original Adventure game. By using text compression techniques, I was able to squeeze the entire game into 48K RAM. The interaction is lightning fast, and nothing ever has to be found on the disk. The whole game is in there: over 130 rooms, 15 treasures, 40 useful objects, and 12 obstacles or opponents.

I will send you a copy FREE! Just send me a blank diskette and postage. Or send $5.00 and I will send the disk and pay the postage. Write to me, Jeff Jacobsen, at Frontier Computing Inc., P. O. Box 402, Logan, Utah 84321.

Problem with QD#5

The first 14 copies that I sent out of Quarterly Disk #5 were incomplete. I forgot to include PMD and FPSUBS. If you have one of those with serial #1 thru #14, send it back; I will add the programs and return it. I'm sorry!

Subscriptions Around the World

We are now sending AAL to over 800 subscribers. Of course most of these are in the U.S.A., but an increasing number are subscribing from other countries. We now have:

15 -- Canada	2 -- Hong Kong
5 -- Sweden	2 -- Ireland
5 -- New Zealand	1 -- Israel
4 -- France	1 -- Italy
4 -- Japan	1 -- Netherlands
3 -- Australia	1 -- Qatar
3 -- England	1 -- Saudi Arabia
3 -- West Germany	1 -- Spain
3 -- South Africa	1 -- Thailand
2 -- Argentina	1 -- Turkey
2 -- Belgium	

And there are also at least a half dozen subscribers with APO addresses, who are stationed in strange exotic lands.

On Dividing by Ten.....Jim Church

Some time ago you asked readers to come up with subroutines to divide by ten (or multiply by one-tenth). I may have come up with the smallest one, although it is certainly not the fastest.

By using SWEET-16 to merely subtract 10 over and over, until the remainder is less than 10, and counting the number of subtractions, I can divide a 16-bit value by ten in a 10-byte subroutine!

In fact, you can divide by any 16-bit value. My program assumes the divisor is in $02,03 and the dividend is in $04,05. These are the Sweet-16 registers 1 and 2. The quotient will be left in $04,05; the remainder will be in $00,01.

I used a copy of Sweet-16 in RAM, from the source code on the S-C Assembler II disk. If you use the copy in the Integer BASIC ROM's, or in the RAM card Integer BASIC, change line 1130 to "SW16 .EQ $F689".

Here is the program listing:

```
                1000 *
                1010 *--------------------------------
                1020 * DIVIDE ANY NUMBER BY 10 OR BY *
                1030 * ANYTHING ELSE FOR THAT MATTER *
                1040 *                               *
                1041 * DIVIDEND - REGISTER 0  $00.01 *
                1042 * DIVISOR  - REGISTER 1  $02.03 *
                1050 * QUOTIENT - REGISTER 2  $04.05 *
                1060 *                               *
                1070 * EXAMPLE---DIVIDE 65534 BY 10  *
                1080 * 00:FE FF 0A 00 00 00 N 300G   *
                1090 *                               *
                1100 *         JIM CHURCH            *
                1110 *--------------------------------
                1120         .OR $300
   9B89-        1130 SW16                .EQ $9B89
                1140 *--------------------------------
                1150 GO
   0300- 20 89 9B 1160         JSR SW16
                1170 STILL.GREATER
   0303- B1     1180         SUB 1        DEDUCT DIVISOR FROM DIVIDEND
   0304- E2     1190         INR 2        ADD 1 TO QUOTIENT
   0305- D1     1200         CPR 1        DIVIDEND > DIVISOR?
   0306- 03 FB  1210         BC  STILL.GREATER
   0308- 00     1220         RTN
   0309- 60     1230         RTS
   000A-        1240 LENGTH .EQ *-GO
                1250 * LOOK IN $00.01 FOR REMAINDER *
```

Apple Assembly Line is published monthly by S-C SOFTWARE, P. O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is $15 per year in the USA, sent Second Class Mail; $18 per year sent First Class Mail in USA, Canada, and Mexico; $28 per year sent Air Mail to other countries. Back issues are available for $1.50 each (other countries add $1 per back issue for postage).